Application Software Security Scanning

Lyle Barner
Jet Propulsion Laboratory, California Institute of Technology
lyle.barner@jpl.nasa.gov

Jet Propulsion Laboratory
California Institute of Technology

Overview

- Motivation
- Overview of pilot program
- Description of target applications
- Scanning techniques
- Overview of results
- Lessons learned
- Path forward and summary

Why is this activity necessary?

- Many projects may not fully consider security aspects during development
  - Not having a well defined process can lead to inconsistent implementation
- Security vulnerabilities uncovered through static analysis and environment scanning can often be a low hanging fruit
  - SQL injection, error message handling, updating to latest version, etc.
- Process must be established

Objectives of the Pilot Program

- Gather metrics
  - Setup time, analysis time, vulnerability metrics, etc.
- Establish a process for performing vulnerability scans
- Gain buy-in from projects and developers

Ultimate Goal

For cybersecurity scans to be included as a regular part of the development lifecycle

Team Members and Roles

- Facilitator: ensure that all experts have the resources they need to complete the scan and disposition the results
- Tools Expert: set up the tool(s), make sure they're properly configured, perform the scans, and post-process the results if necessary
- Source Code Expert: work with the cybersecurity expert to disposition the results of the automated and manual scans
- Cybersecurity Expert: review the results from the automated scans and perform the manual scans based on their expert opinion

Overview of Scanning Process

1. Identify interested parties and target project
2. Select, install, and configure scanning tools
3. Perform scans of codebase and operating environment
4. Review results with source code expert and cybersecurity expert
5. Collect metrics
6. Make code changes as necessary using data uncovered during scans

Overview of Pilot Subjects

- 2 active ground software projects
  - Written in Java
  - Roughly 30k lines of code
- Motivated teams that want to get an accurate picture of the security concerns in their codebase
- Projects requested to remain anonymous due to the sensitive nature of the findings

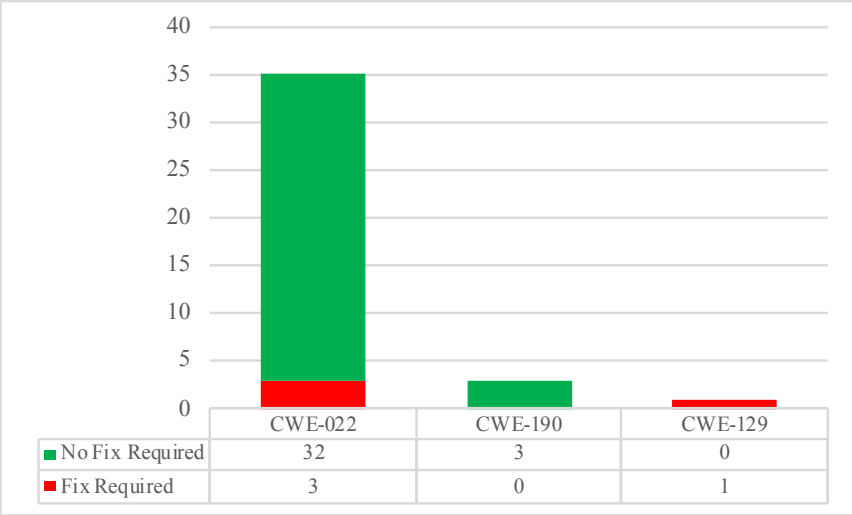
Overview of Scanning Tools and Techniques

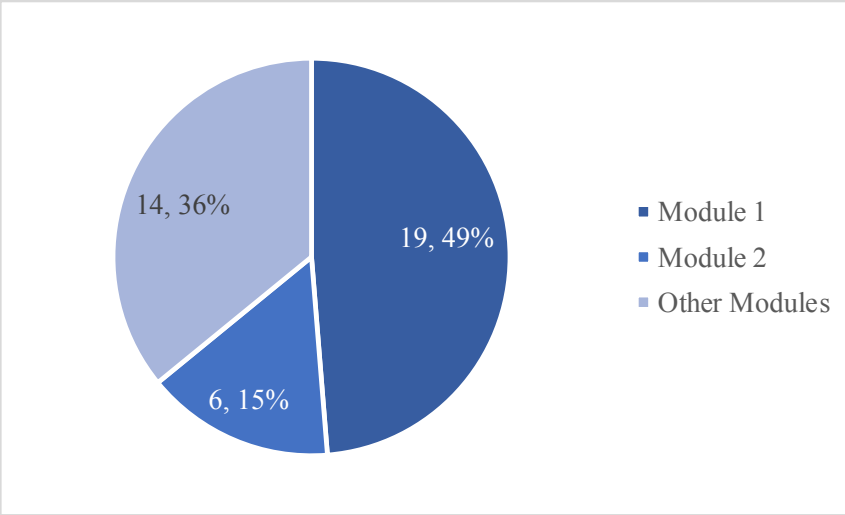
- Tool 1: Static analysis tool used for scanning the source code for vulnerabilities. Supports 107 CWEs for Java
- Tool 2: Additional static analysis tool, different than Tool 1. Supports 618 CWEs for Java
- Tool 3: Tool used for scanning the operating environment to identify potential vulnerabilities
- Manual Scan: Performed by cybersecurity expert based on their expert opinion

Scanning Results: Project 1

Metric	Value
Language	Java
Lines of code	24,310
Total findings	39
Finding rate	1.61/thousand lines of code
True defect rate	0.16/thousand lines of code
Findings requiring a code change	12.5%
Analysis time	Tools setup: 5-6 hrs SCE analysis: 2-3 hrs CSE analysis: 2 hrs Total: 9-11 hrs

- Valid issues uncovered and actionable information provided
- Results of manual scanning
  - Issues concerning configuration of third-party software

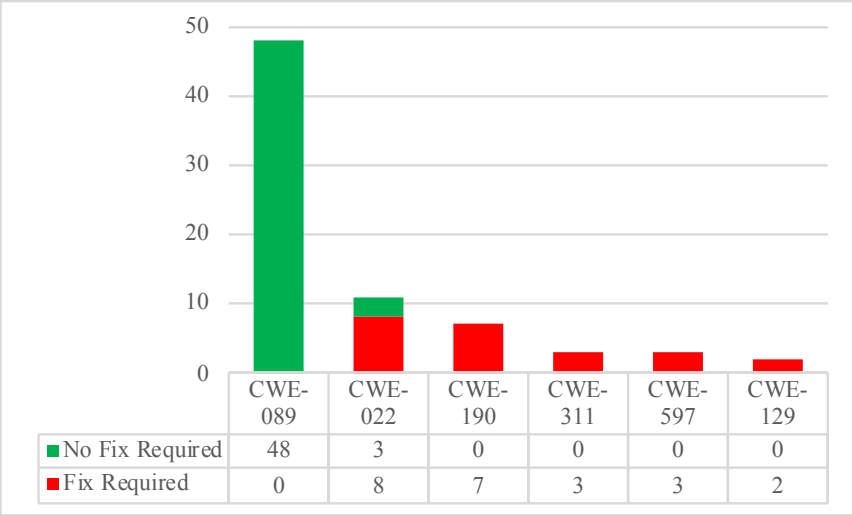


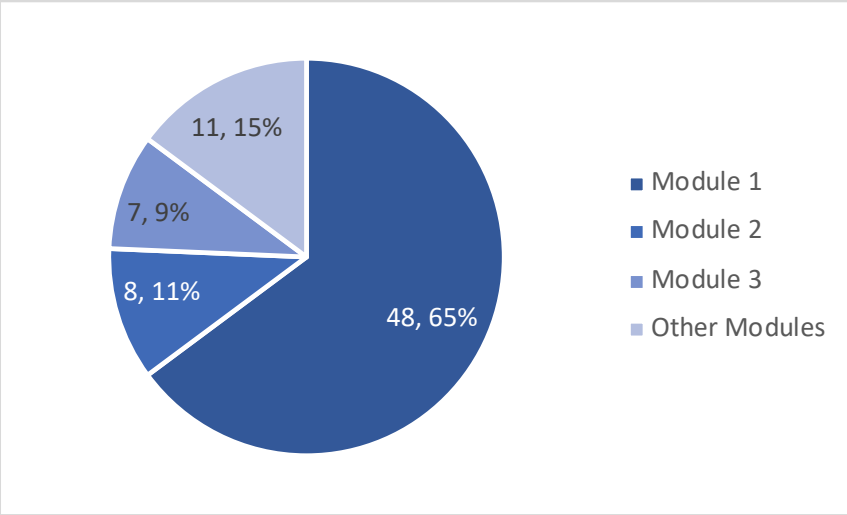


Scanning Results: Project 2

Metric	Value
Language	Java
Lines of code	36,725
Total findings	74
Finding rate	2.01/thousand lines of code
True defect rate	1.39/thousand lines of code
Findings requiring a code change	69%
Analysis time	Tools setup: 2-3 hrs Lead analysis: 2-3 hrs SCE analysis: 4-5 hrs CSE analysis: 4 hrs Total: 12-15 hrs

- Valid issues uncovered and actionable information provided
- Results of manual scanning
  - Issues with documentation uncovered during manual cybersecurity expert analysis
  - Issues concerning configuration of third-party software





Comparison of Tool 1 and Tool 2

- Tool 2 has a much larger number covered CWEs and results for Project 2
  - Overlap of 81 covered CWEs between Tool 1 and Tool 2
- There was almost no overlap in the warnings that wee identified by both tools for any given CWE
  - 17 cases where there was no concurrence between tools
  - 63 cases where there was potential concurrence (no warnings)
  - 1 case of partial concurrence
  - 0 cases of identical results

	Tool 1	Tool 2
Covered CWEs	107	618
Total Warnings	74	2795
Warning Concurrence	1	

Examples of CWE Warnings Identified

- **CWE-022**: “Improper limitation of a pathname to a restricted directory”
- **CWE-311**: “Missing encryption of sensitive data”
- **CWE-129**: “Improper validation of array index”
- Source for more information about CWEs
  - https://cwe.mitre.org

Observations

- Types of vulnerabilities found are based on functionality
- Vulnerabilities are not evenly distributed
  - Grouped based on functionality of the code
- Analysis time depends heavily on the types/distribution of warnings

Lessons Learned

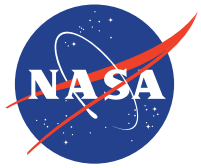
- There is relatively little cost associated with performing these types of scans
- Proper configuration of the analyzer is crucial
- There is very little overlap between the different static analysis tools
- The operating environment and codebase must both be examined
- Most of the information needed to perform scans is readily available
- Special measures need to be taken when sharing results
- Types of vulnerabilities found are based on functionality, not size of codebase
- Vulnerabilities are often not even distributed throughout the codebase
- Grouping warnings into categories is helpful for dispositioning
- A cybersecurity expert and knowledgeable developer must work together to make an accurate assessment of the warnings
- True positives can be used to create a rolling list of design rules
- A formalized process must be developed for analyzing and prioritizing warnings

Path Forward

- Begin developing formal scanning processes and procedures
- Explore use of other scanning tools
- Review and further refine processes with other projects
- Incorporate new processes in standard development processes as part of the application security assurance lifecycle for both in-house development and JPL suppliers
- Develop training and exposure materials for development teams

Summary

- 4 primary roles required: facilitator, tools expert, source code expert, and cybersecurity expert
- A process is being developed and refined
- Specific tools were used, but others are being investigated
- Analysis metrics were gathered to inform formal process development
- Lab-wide roll out will be planned after formal process is developed and refined
  - Currently support projects that are interested in performing these kinds of scans



Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov